BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 06012467 A

(43) Date of publication of application: 21 . 01 . 94

(51) Int. Cl G06F 15/60

(21) Application number: 04167727

(22) Date of filing: 25 . 06 . 92

(71) Applicant: SONY CORP

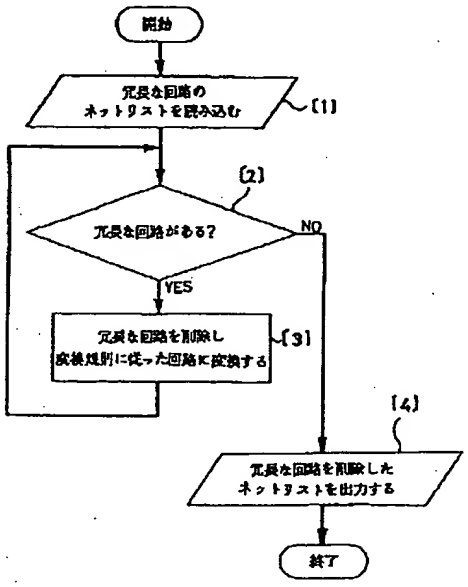
(72) Inventor: OKI MITSUHARU

(54) LOGIC CIRCUIT COMPRESSING METHOD

(57) Abstract:

PURPOSE: To realize an LSI having a small circuit scale by deleting redundant circuits from a network list of redundant circuit constitutions to generate a new network list and designing layout from this new network list.

CONSTITUTION: In a step 1, the network list of redundant circuits is read in. In a step 2, it is discriminated whether a specific redundant circuit exists in the network list or not. If it exists there, the redundant circuit is deleted in accordance with a conversion rule in a step 3. Steps 2 and 3 are repeated until redundant circuits are completely deleted from the network list. When redundant circuits are completely deleted from the network list in the step 2, the new list of circuits other than redundant circuits is outputted in a step 4.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平6－12467

(43)公開日　平成６年(1994)１月21日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　15/60	３６０　Ｋ	7922－5Ｌ		

審査請求　未請求　請求項の数７(全 14 頁)

(21)出願番号　特願平4－167727

(22)出願日　平成４年(1992)６月25日

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号

(72)発明者　大木　光晴
東京都品川区北品川６丁目７番35号　ソニー株式会社内

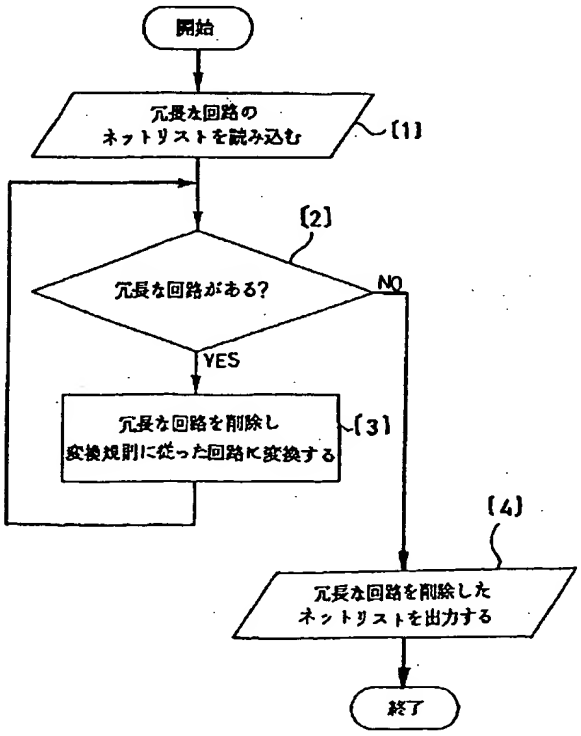
(74)代理人　弁理士　松隈　秀盛

(54)【発明の名称】　論理回路圧縮方法

(57)【要約】

【目的】　冗長な回路構成のネットリストから冗長な回路を削除したネットリストを作り、このような新たなネットリストからレイアウト設計することにより回路規模の小さいＬＳＩを実現する。

【構成】　ステップ〔１〕で冗長な回路のネットリストが読み込まれる。次にステップ〔２〕でそのネットリスト内に特定の冗長な回路があるかないかが判断される。ここでもし冗長な回路があればステップ〔３〕で変換規則に従って冗長な回路が削除される。さらにこのステップ〔２〕及び〔３〕が冗長な回路がなくなるまで繰り返し実行される。そしてステップ〔２〕でそのネットリスト内にある冗長な回路がなくなると、ステップ〔４〕で冗長でない回路のネットリストが出力される。



【特許請求の範囲】

【請求項１】　少なくとも１つの入力信号は定数である加算器若しくは単位遅延素子を含む冗長な回路構成のネットリストを入力とし、

特定の変換規則によって冗長でない回路構成のネットリストを出力するようにしたことを特徴とする論理回路圧縮方法。

【請求項２】　上記特定の変換規則は、

３つの入力とも常に０である全加算器は削除し、全加算器の出力であるキャリアウト端子の配線に０を供給させ、全加算器の出力であるサム端子の配線に０を供給させる、

２つの入力が常に０であり、残る１つの入力が常に１である全加算器は削除し、全加算器の出力であるキャリアウト端子の配線に０を供給させ、全加算器の出力であるサム端子の配線に１を供給させる、

１つの入力が常に０であり、残る２つの入力が常に１である全加算器は削除し、全加算器の出力であるキャリアウト端子の配線に１を供給させ、全加算器の出力であるサム端子の配線に０を供給させる、

３つの入力とも常に１である全加算器は削除し、全加算器の出力であるキャリアウト端子の配線に１を供給させ、全加算器の出力であるサム端子の配線に１を供給させる、という変換の少なくとも一つを含んでいることを特徴とする請求項１に記載の論理回路圧縮方法。

【請求項３】　上記特定の変換規則は、

２つの入力が常に０であり、残る１つの入力が乗数、被乗数の値によって変化する全加算器は削除し、全加算器の出力であるキャリアウト端子の配線に０を供給させ、全加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する値を供給させる、

１つの入力が常に０であり、もう１つの入力が常に１であり、残る１つの入力が乗数、被乗数の値によって変化する全加算器は削除し、全加算器の出力であるキャリアウト端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、全加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給させる、

２つの入力が常に１であり、残る１つの入力が乗数、被乗数の値によって変化する全加算器は削除し、全加算器の出力であるキャリアウト端子の配線に１を供給させ、全加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する値を供給させる、という変換の少なくとも一つを含んでいることを特徴とする請求項１に記載の論理回路圧縮方法。

【請求項４】　上記特定の変換規則は、

１つの入力が常に０であり、残る２つの入力が乗数、被乗数の値によって変化する全加算器は削除し、半加算器の２つの入力端子に上記乗数、被乗数の値によって変化する２つの値を供給させ、全加算器の出力であるキャリアウト端子の配線に上記半加算器の出力であるキャリアウトを供給させ、全加算器の出力であるサム端子の配線に上記半加算器の出力であるサムを供給させる、

１つの入力が常に１であり、残る２つの入力が乗数、被乗数の値によって変化する全加算器は削除し、全加算器の出力であるキャリアウト端子の配線に上記乗数、被乗数の値によって変化する２つの値の論理和を供給させ、全加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する２つの値の排他的否定論理和を供給させる、という変換の少なくとも一つを含んでいることを特徴とする請求項１に記載の論理回路圧縮方法。

【請求項５】　上記特定の変換規則は、

２つの入力とも常に０である半加算器は削除し、半加算器の出力であるキャリアウト端子の配線に０を供給させ、半加算器の出力であるサム端子の配線に０を供給させる、

１つの入力が常に０であり、残る１つの入力が常に１である半加算器は削除し、半加算器の出力であるキャリアウト端子の配線に０を供給させ、半加算器の出力であるサム端子の配線に１を供給させる、

２つの入力とも常に１である半加算器は削除し、半加算器の出力であるキャリアウト端子の配線に１を供給させ、半加算器の出力であるサム端子の配線に０を供給させる、という変換の少なくとも一つを含んでいることを特徴とする請求項１に記載の論理回路圧縮方法。

【請求項６】　上記特定の変換規則は、１つの入力が常に０であり、残る１つの入力が乗数、被乗数の値によって変化する半加算器は削除し、半加算器の出力であるキャリアウト端子の配線に０を供給させ、半加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する値を供給させる、

１つの入力が常に１であり、残る１つの入力が乗数、被乗数の値によって変化する半加算器は削除し、半加算器の出力であるキャリアウト端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、半加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給させる、という変換の少なくとも一つを含んでいることを特徴とする請求項１に記載の論理回路圧縮方法。

【請求項７】　上記特定の変換規則は、

単位遅延素子に定数が入力されているときは、その単位遅延素子を削除し、その単位遅延素子の出力端子に上記定数を供給させる、という変換を含んでいることを特徴とする請求項１に記載の論理回路圧縮方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、ディジタル信号処理回路をＬＳＩ化する場合等に使用される論理回路圧縮方法に関するものである。

【０００２】

【従来の技術】ディジタル信号処理回路をＬＳＩ化する場合、論理設計とレイアウト設計の２段階により設計が行われる。

【０００３】論理設計とは、アンドゲートやオアゲートやフルアダー（全加算器）やハーフアダー（半加算器）などと、それらゲートの接続をする配線の情報を持ったデータファイルを作成することである。一般的に、このデータファイルは、ネットリストと呼ばれている。ネットリストは、各半導体工場の専用のフォーマットのものや、共通化されたフォーマット（例えば、ＥＤＩＦ、Ｖerilog－ＨＤＬ、ＶＨＤＬなど）のものがある。

【０００４】レイアウト設計とは、上述のネットリストのデータを受け取り、実際のＬＳＩチップ上のどの位置にどのゲートやどの配線を置くかということを決める作業である。

【０００５】論理設計について、さらに具体例を用いて説明する。

【０００６】図９のＡは、

乗数　：$X=(X_7,X_6,X_5,X_4,X_3,X_2,X_1,X_0)$　8bit

被乗数：$Y=(Y_4,Y_3,Y_2,Y_1,Y_0)$　5bit

の乗算をして、

出力　：$P=(P_{12},P_{11},P_{10},P_9,P_8,P_7,P_6,P_5,P_4,P_3,P_2,P_1,P_0)$　13bit

を得るための計算方法である。

【０００７】但し、

$A_i=Y_0$とX_iのアンド（論理積）

$B_i=Y_1$とX_iのアンド（論理積）

$C_i=Y_2$とX_iのアンド（論理積）

$D_i=Y_3$とX_iのアンド（論理積）

$E_i=Y_4$とX_iのアンド（論理積）　（ｉ＝０～７）

である。

【０００８】即ち、部分積A_i、B_i、C_i、D_i、E_iを図９のＡに示すように加算することにより、乗算結果P_iを得ることが出来る。

【０００９】さて図９のＡの計算は全加算器や半加算器により計算していくことが可能であり、結局、論理設計は全加算器や半加算器、そしてそれらゲートの接続をする配線情報を作成することになる。

【００１０】しかし上述の計算では、部分積A_i、B_i、C_i、D_i、E_iは桁が１ビットずつシフトしており、規則的ではないため、ネットリスト作成は簡単でない（欠点１）。また仕様が変更されて、８ビット×５ビットではなく、８ビット×６ビットとなった時、最初からネットリストを書き直さなくてはいけないなどの欠点があった（欠点２）。

【００１１】さらに、本願出願人が先に提案した回路記述合成装置（特願平３－１９５７１４号）では、パラメトリックに回路を記述できる装置について述べられているが、サイズ変換部での変換操作が難しい。なぜなら各部分積が１ビットずつシフトしており、規則的でなく、サイズ変換のアルゴリズムが複雑な為である（欠点３）。

【００１２】これら欠点１、２、３は、部分積同士で規則性がないため起こる。そこでこれらの欠点を克服する方法として、図９のＢの計算を行う事が考えられる。

【００１３】図９のＢの計算は、図９のＡの部分積の空白個所に０を挿入して、１３ビットのデータ

$(0,0,0,0,A_7,A_6,A_5,A_4,A_3,A_2,A_1,A_0)$

$(0,0,0,0,B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0,0)$

$(0,0,0,C_7,C_6,C_5,C_4,C_3,C_2,C_1,C_0,0,0)$

$(0,0,D_7,D_6,D_5,D_4,D_3,D_2,D_1,D_0,0,0,0)$

$(0,E_7,E_6,E_5,E_4,E_3,E_2,E_1,E_0,0,0,0,0)$

を部分積として、加算をする方法を取っている。これにより、加算すべき部分積が規則正しくなる。

【００１４】この図９のＢの計算をする回路図が図１０である。図１０の第１段の全加算器（ＦＡ）群により、３つの部分積、

$(0,0,0,0,0,A_7,A_6,A_5,A_4,A_3,A_2,A_1,A_0)$

$(0,0,0,0,B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0,0)$

$(0,0,0,C_7,C_6,C_5,C_4,C_3,C_2,C_1,C_0,0,0)$

が加算され、第２段の全加算器（ＦＡ）群により、部分積、

$(0,0,D_7,D_6,D_5,D_4,D_3,D_2,D_1,D_0,0,0,0)$

が加算され、第３段の全加算器（ＦＡ）群により、さらに部分積、

$(0,E_7,E_6,E_5,E_4,E_3,E_2,E_1,E_0,0,0,0,0)$

が加算される。

【００１５】第１～３段の全加算器（ＦＡ）群では、キャリセーブ方式で加算しているので、最後に、桁上げ加算をする必要があり、その計算を行うのが最終段の全加算器（ＦＡ）群である。

【００１６】図１０に対応したネットリストは、１３×３個の全加算器（ＦＡ）群と、全加算器（ＦＡ）同士の接続をする配線情報、及び、１３個の全加算器（ＦＡ）群より成る桁上げ加算器より構成される。

【００１７】１３×３個の全加算器（ＦＡ）群（第１～３段の全加算器（ＦＡ）群）が、規則正しく配列されていることが図１０より明らかであり、当然、そのネットリストも規則正しい。従って、ネットリスト作成は簡単である（欠点１の克服）。

【００１８】また仕様が変更されて、８ビット×５ビットではなく、８ビット×６ビットとなった時、ネットリストに、第１～３段の全加算器（ＦＡ）群と同一の１３個の全加算器（ＦＡ）群を第４の全加算器（ＦＡ）群として追加するだけで良い（欠点２の克服）。

【００１９】さらに回路記述合成装置のサイズ変換部での変換操作が簡単である。なぜなら、被乗数分の全加算器（ＦＡ）群だけを作成するという単純な操作で良いからである（欠点３の克服）。

【００２０】このように、ダミーデータ０を挿入することにより、部分積に規則性を持たせて論理設計を容易化できる。

【００２１】しかしこれから作られるネットリストを用いて、レイアウト設計を行うと、図１０に示す回路と同じ構成の回路が、ＬＳＩチップ上に作られる。つまり０同士の加算を行うという無意味な全加算器などもチップ上に作られてしまい、回路規模が大きくなってしまうという欠点があった。

【００２２】

【発明が解決しようとする課題】問題点は、従来の方法では論理設計を容易化するためなどにネットリスト中には冗長な回路が含まれていたことにより、実際に作られるＬＳＩの回路規模が大きくなってしまうというものである。

【００２３】

【課題を解決するための手段】本発明による第１の手段は、少なくとも１つの入力信号は定数である加算器若しくは単位遅延素子を含む冗長な回路構成のネットリストを入力（ステップ〔１〕）とし、特定の変換規則（ステップ〔２〕及び〔３〕）によって冗長でない回路構成のネットリストを出力（ステップ〔４〕）するようにしたことを特徴とする論理回路圧縮方法である。

【００２４】本発明による第２の手段は、上記特定の変換規則は、３つの入力とも常に０である全加算器(Full Adder)は削除し、全加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、全加算器の出力であるサム（Sum）端子の配線に０を供給させる（図２のＡ）、２つの入力が常に０であり、残る１つの入力が常に１である全加算器(Full Adder)は削除し、全加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、全加算器の出力であるサム（Sum）端子の配線に１を供給させる（図２のＢ）、１つの入力が常に０であり、残る２つの入力が常に１である全加算器(Full Adder)は削除し、全加算器の出力であるキャリアウト（Co）端子の配線に１を供給させ、全加算器の出力であるサム（Sum）端子の配線に０を供給させる（図２のＣ）、３つの入力とも常に１である全加算器(Full Adder)は削除し、全加算器の出力であるキャリアウト（Co）端子の配線に１を供給させ、全加算器の出力であるサム（Sum）端子の配線に１を供給させる（図２のＤ）、という変換の少なくとも一つを含んでいることを特徴とする第１の手段に記載の論理回路圧縮方法である。

【００２５】本発明による第３の手段は、上記特定の変換規則は、２つの入力が常に０であり、残る１つの入力（in1）が乗数、被乗数の値によって変化する全加算器(Full Adder)は削除し、全加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、全加算器の出力であるサム（Sum）端子の配線に上記乗数、被乗数の値によって変化する値を供給させる（図３のＡ）、１つの入力が常に０であり、もう１つの入力が常に１であり、残る１つの入力（in1）が乗数、被乗数の値によって変化する全加算器(Full Adder)は削除し、全加算器の出力であるキャリアウト（Co）端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、全加算器の出力であるサム（Sum）端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給させる（図３のＢ）、２つの入力が常に１であり、残る１つの入力（in1）が乗数、被乗数の値によって変化する全加算器(Full Adder)は削除し、全加算器の出力であるキャリアウト（Co）端子の配線に１を供給させ、全加算器の出力であるサム（Sum）端子の配線に上記乗数、被乗数の値によって変化する値を供給させる（図３のＣ）、という変換の少なくとも一つを含んでいることを特徴とする第１の手段に記載の論理回路圧縮方法である。

【００２６】本発明による第４の手段は、上記特定の変換規則は、１つの入力が常に０であり、残る２つの入力（in1、in2）が乗数、被乗数の値によって変化する全加算器(Full Adder)は削除し、半加算器(Half Adder)の２つの入力端子に上記乗数、被乗数の値によって変化する２つの値を供給させ、全加算器の出力であるキャリアウト（Co）端子の配線に上記半加算器の出力であるキャリアウトを供給させ、全加算器の出力であるサム（Sum）端子の配線に上記半加算器の出力であるサムを供給させる（図４のＡ）、１つの入力が常に１であり、残る２つの入力（in1、in2）が乗数、被乗数の値によって変化する全加算器(Full Adder)は削除し、全加算器の出力であるキャリアウト（Co）端子の配線に上記乗数、被乗数の値によって変化する２つの値の論理和（オア回路）を供給させ、全加算器の出力であるサム（Sum）端子の配線に上記乗数、被乗数の値によって変化する２つの値の排他的否定論理和（イクスクルーシブノア回路）を供給させる（図４のＢ）、という変換の少なくとも一つを含んでいることを特徴とする第１の手段に記載の論理回路圧縮方法である。

【００２７】本発明による第５の手段は、上記特定の変

換規則は、２つの入力とも常に０である半加算器(Half Adder)は削除し、半加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、半加算器の出力であるサム（Sum ）端子の配線に０を供給させる（図５のＡ）、１つの入力が常に０であり、残る１つの入力が常に１である半加算器(Half Adder)は削除し、半加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、半加算器の出力であるサム（Sum ）端子の配線に１を供給させる（図５のＢ）、２つの入力とも常に１である半加算器(Half Adder)は削除し、半加算器の出力であるキャリアウト（Co）端子の配線に１を供給させ、半加算器の出力であるサム（Sum ）端子の配線に０を供給させる（図５のＣ）、という変換の少なくとも一つを含んでいることを特徴とする第１の手段に記載の論理回路圧縮方法である。

【００２８】本発明による第６の手段は、上記特定の変換規則は、１つの入力が常に０であり、残る１つの入力（in1 ）が乗数、被乗数の値によって変化する半加算器(Half Adder)は削除し、半加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、半加算器の出力であるサム（Sum ）端子の配線に上記乗数、被乗数の値によって変化する値を供給させる（図６のＡ）、１つの入力が常に１であり、残る１つの入力（in1 ）が乗数、被乗数の値によって変化する半加算器(Half Adder)は削除し、半加算器の出力であるキャリアウト（Co）端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、半加算器の出力であるサム（Sum）端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給させる（図６のＢ）、という変換の少なくとも一つを含んでいることを特徴とする第１の手段に記載の論理回路圧縮方法である。

【００２９】本発明による第７の手段は、上記特定の変換規則は、単位遅延素子に定数が入力されているときは、その単位遅延素子を削除し、その単位遅延素子の出力端子に上記定数を供給させる（図７）、という変換を含んでいることを特徴とする第１の手段に記載の論理回路圧縮方法である。

【００３０】

【作用】これによれば、冗長な回路構成のネットリストから冗長な回路を削除したネットリストを作ることができ、このような新たなネットリストからレイアウト設計することにより回路規模の小さいＬＳＩを実現することができる。

【００３１】

【実施例】本発明の論理回路圧縮方法のフローチャートの例を図１に示す。

【００３２】すなわちこの図において、ステップ〔１〕で冗長な回路のネットリストが読み込まれる。次にステップ〔２〕でそのネットリスト内に後述する特定の冗長な回路があるかないかが判断される。ここでもし冗長な回路があればステップ〔３〕で後述する変換規則に従って冗長な回路が削除される。

【００３３】さらにこのステップ〔２〕及び〔３〕が冗長な回路がなくなるまで繰り返し実行される。そしてステップ〔２〕でそのネットリスト内にある冗長な回路がなくなると、ステップ〔４〕で冗長でない回路のネットリストが出力される。

【００３４】こうして上述の方法によれば、冗長な回路構成のネットリストから冗長な回路を削除したネットリストを作ることができ、このような新たなネットリストからレイアウト設計することにより回路規模の小さいＬＳＩを実現することができるものである。

【００３５】さらに上述の方法で、冗長な回路の有無を判断して削除するための特定の変換規則には、以下のようなものが用いられる。

【００３６】すなわち図２のＡは、３つの入力とも常に０である全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、全加算器の出力であるサム（Sum ）端子の配線に０を供給させる。

【００３７】また図２のＢは、２つの入力が常に０であり、残る１つの入力が常に１である全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、全加算器の出力であるサム（Sum ）端子の配線に１を供給させる。

【００３８】また図２のＣは、１つの入力が常に０であり、残る２つの入力が常に１である全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアウト（Co）端子の配線に１を供給させ、全加算器の出力であるサム（Sum ）端子の配線に０を供給させる。

【００３９】また図２のＤは、３つの入力とも常に１である全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアウト（Co）端子の配線に１を供給させ、全加算器の出力であるサム（Sum ）端子の配線に１を供給させる。

【００４０】さらに図３のＡは、２つの入力が常に０であり、残る１つの入力（in1 ）が乗数、被乗数の値によって変化する全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、全加算器の出力であるサム（Sum）端子の配線に上記乗数、被乗数の値によって変化する値を供給させる。

【００４１】また図３のＢは、１つの入力が常に０であり、もう１つの入力が常に１であり、残る１つの入力（in1 ）が乗数、被乗数の値によって変化する全加算器

(FullAdder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアウト（Co）端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、全加算器の出力であるサム（Sum ）端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給させる。

【００４２】また図３のＣは、２つの入力が常に１であり、残る１つの入力（in1 ）が乗数、被乗数の値によって変化する全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアウト（Co）端子の配線に１を供給させ、全加算器の出力であるサム（Sum ）端子の配線に上記乗数、被乗数の値によって変化する値を供給させる。

【００４３】さらに図４のＡは、１つの入力が常に０であり、残る２つの入力（in1 、in2）が乗数、被乗数の値によって変化する全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように半加算器(Half Adder)の２つの入力端子に上記乗数、被乗数の値によって変化する２つの値を供給させ、全加算器の出力であるキャリアウト（Co）端子の配線に上記半加算器の出力であるキャリアウトを供給させ、全加算器の出力であるサム（Sum ）端子の配線に上記半加算器の出力であるサムを供給させる。

【００４４】また図４のＢは、１つの入力が常に１であり、残る２つの入力（in1 、in2 ）が乗数、被乗数の値によって変化する全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアウト（Co）端子の配線に上記乗数、被乗数の値によって変化する２つの値の論理和（オア回路）を供給させ、全加算器の出力であるサム（Sum ）端子の配線に上記乗数、被乗数の値によって変化する２つの値の排他的否定論理和（イクスクルーシブノア回路）を供給させる。

【００４５】さらに図５のＡは、２つの入力とも常に０である半加算器(Half Adder)の場合であって、この場合にはこの半加算器は削除し、同図の右側に示すように半加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、半加算器の出力であるサム（Sum ）端子の配線に０を供給させる。

【００４６】また図５のＢは、１つの入力が常に０であり、残る１つの入力が常に１である半加算器(Half Adder)の場合であって、この場合にはこの半加算器は削除し、同図の右側に示すように半加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、半加算器の出力であるサム（Sum ）端子の配線に１を供給させる。

【００４７】また図５のＣは、２つの入力とも常に１である半加算器(Half Adder)の場合であって、この場合にはこの半加算器は削除し、同図の右側に示すように半加算器の出力であるキャリアウト（Co）端子の配線に１を供給させ、半加算器の出力であるサム（Sum ）端子の配線に０を供給させる。

【００４８】さらに図６のＡは、１つの入力が常に０であり、残る１つの入力（in1 ）が乗数、被乗数の値によって変化する半加算器(Half Adder)の場合であって、この場合にはこの半加算器は削除し、同図の右側に示すように半加算器の出力であるキャリアウト（Co）端子の配線に０を供給させ、半加算器の出力であるサム（Sum）端子の配線に上記乗数、被乗数の値によって変化する値を供給させる。

【００４９】また図６のＢは、１つの入力が常に１であり、残る１つの入力（in1 ）が乗数、被乗数の値によって変化する半加算器(Half Adder)の場合であって、この場合にはこの半加算器は削除し、同図の右側に示すように半加算器の出力であるキャリアウト（Co）端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、半加算器の出力であるサム（Sum ）端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給させる。

【００５０】さらに図７は、単位遅延素子（フリップフロップＦＦ）に定数が入力されている場合であって、この場合には、その単位遅延素子を削除し、同図の右側に示すようにその単位遅延素子の出力端子に上記定数を供給させる。

【００５１】これによって、例えば上述の図１０に示した冗長な回路構成のネットリストから例えば図８に示すような冗長な回路を削除したネットリストを作ることができ、このような新たなネットリストからレイアウト設計することにより回路規模の小さいＬＳＩを実現することができる。

【００５２】なお上述の例では、いわゆるキャリセーブ型の乗算回路について説明をしたが、本発明はこの他にも例えばWallace Tree（ワラス・ツリー）型で部分積を加算していく乗算回路や、さらに乗算回路以外の回路についても適用できる。

【００５３】また本発明の論理回路圧縮方法によれば、以下に挙げる利点がある。即ち、論理設計の段階では設計容易化の為に規則性を重んじて冗長な回路が含まれた形で設計していきネットリストを作成する。そのネットリストに対して本発明の論理圧縮プログラムを適用して冗長でない回路のネットリストに変換する。そしてこの変換されたネットリストによりレイアウト設計を行う。これにより論理設計は容易になるとともに、冗長でない回路のレイアウトを行えるので実際のＬＳＩの回路規模も小さくて済むものである。

【００５４】

【発明の効果】この発明によれば、冗長な回路構成のネットリストから冗長な回路を削除したネットリストを作ることができ、このような新たなネットリストからレイアウト設計することにより回路規模の小さいＬＳＩを実

現することができるようになった。

【図面の簡単な説明】

【図１】本発明による論理回路圧縮方法の一例のフローチャート図である。

【図２】本発明による特定の変換規則の説明のための図である。

【図３】本発明による特定の変換規則の説明のための図である。

【図４】本発明による特定の変換規則の説明のための図である。

【図５】本発明による特定の変換規則の説明のための図である。

【図６】本発明による特定の変換規則の説明のための図である。

【図７】本発明による特定の変換規則の説明のための図である。

【図８】本発明を用いて冗長な回路を削除した部分積加算回路の一例の構成図である。

【図９】乗算計算の説明のための図である。

【図１０】従来の部分積加算回路の構成図である。

【符号の説明】

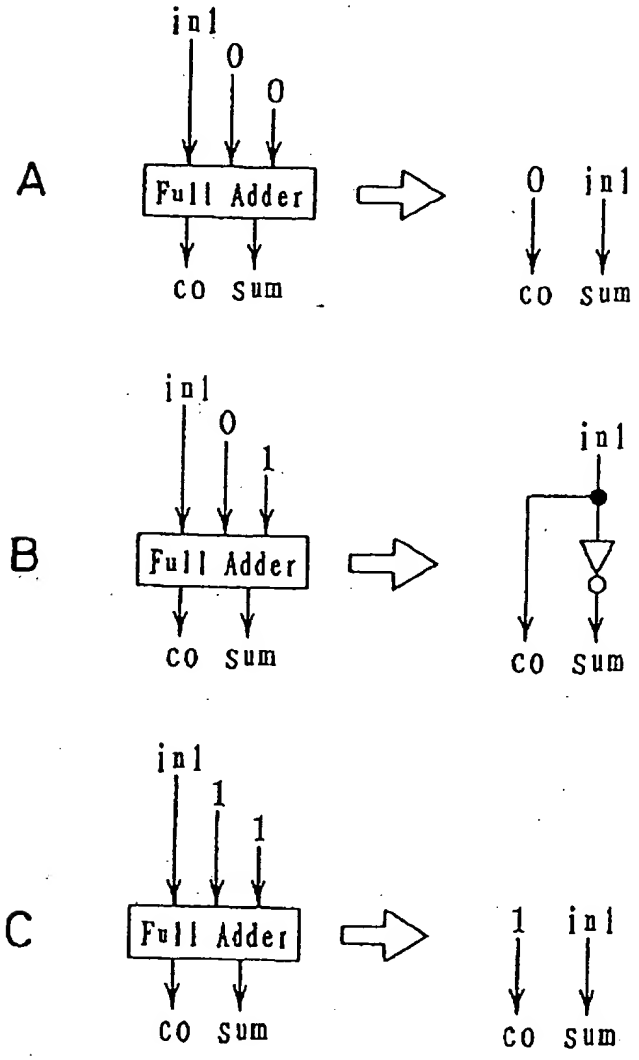
〔１〕　冗長な回路のネットリストの読み込みのステップ

〔２〕　ネットリスト内の特定の冗長な回路の有無の判断のステップ

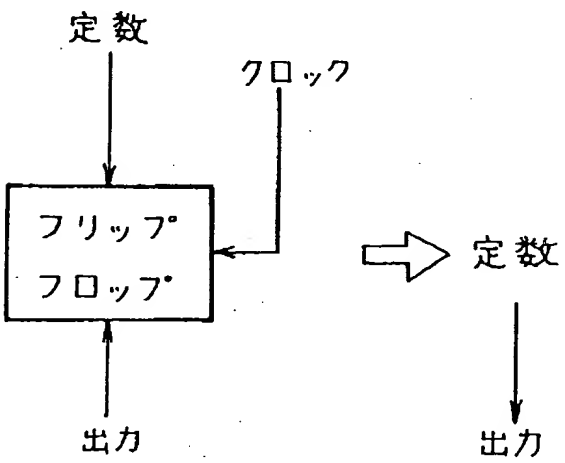
〔３〕　変換規則に従って冗長な回路を削除するステップ

〔４〕　冗長でない回路のネットリストを出力するステップ

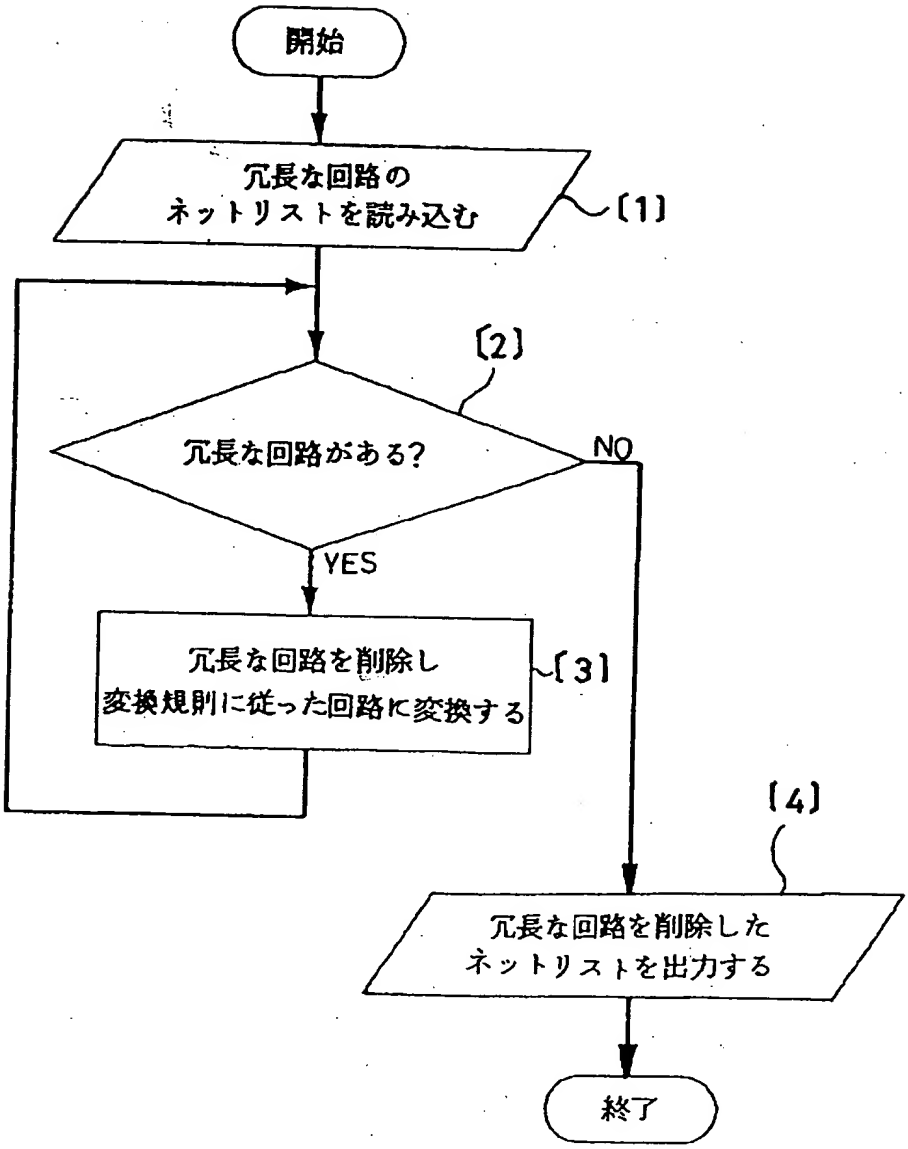
【図３】



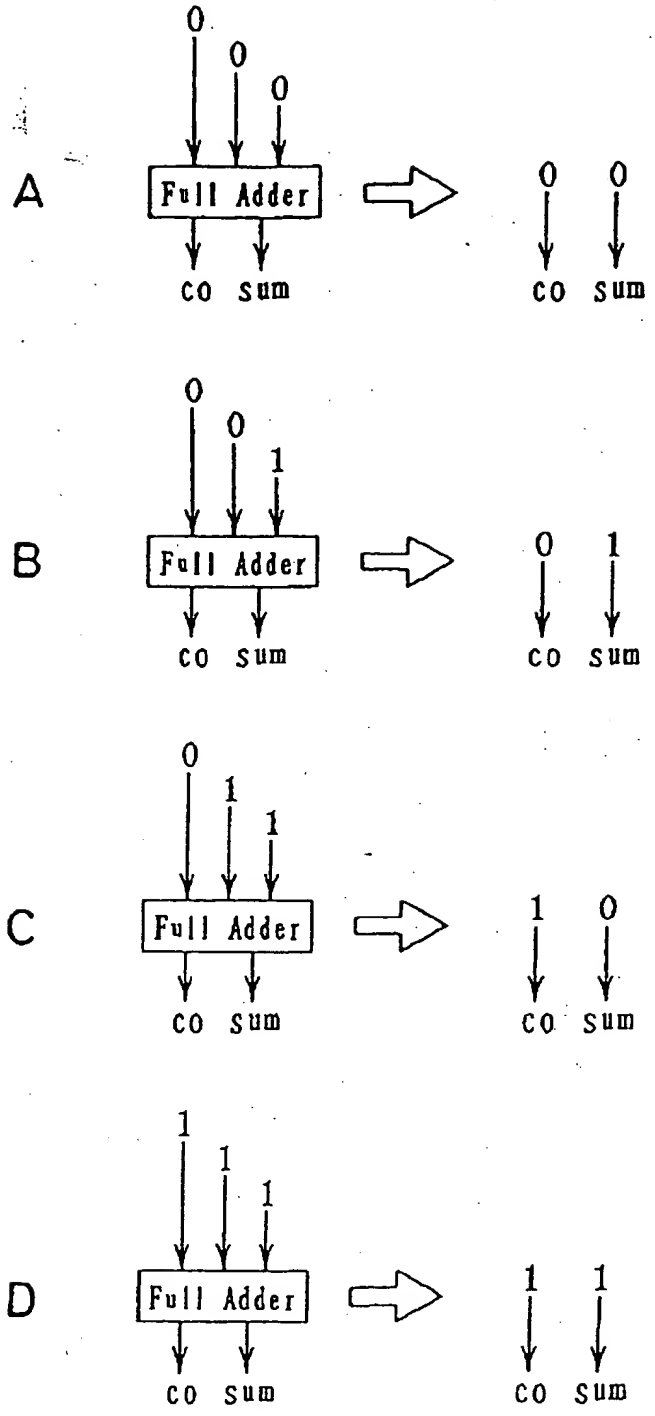
【図７】



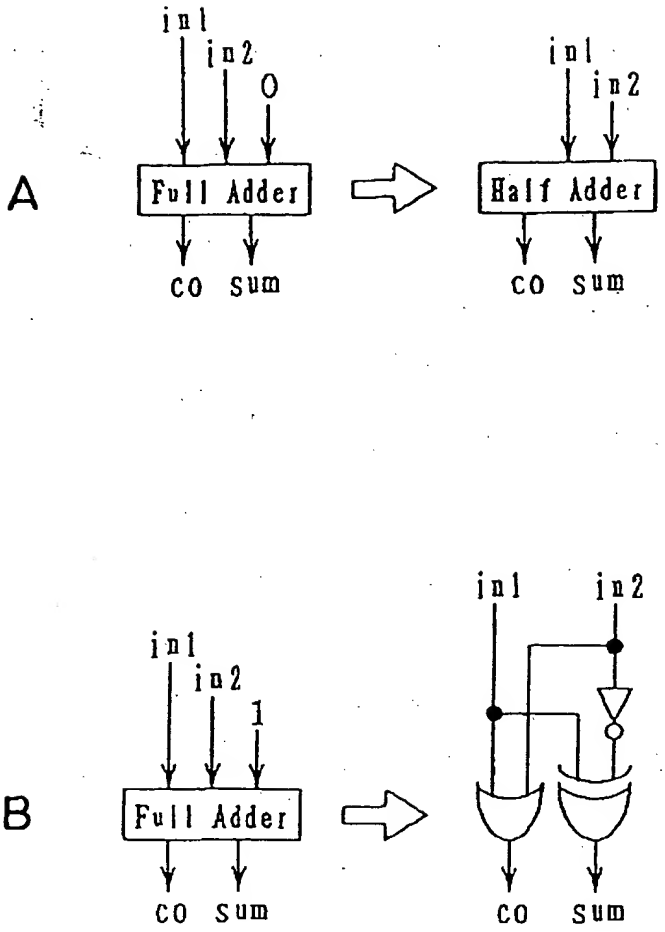
【図1】


開始
冗長な回路の
ネットリストを読み込む
〔1〕
〔2〕
冗長な回路がある?
NO
YES
冗長な回路を削除し
変換規則に従った回路に変換する
〔3〕
〔4〕
冗長な回路を削除した
ネットリストを出力する
終了

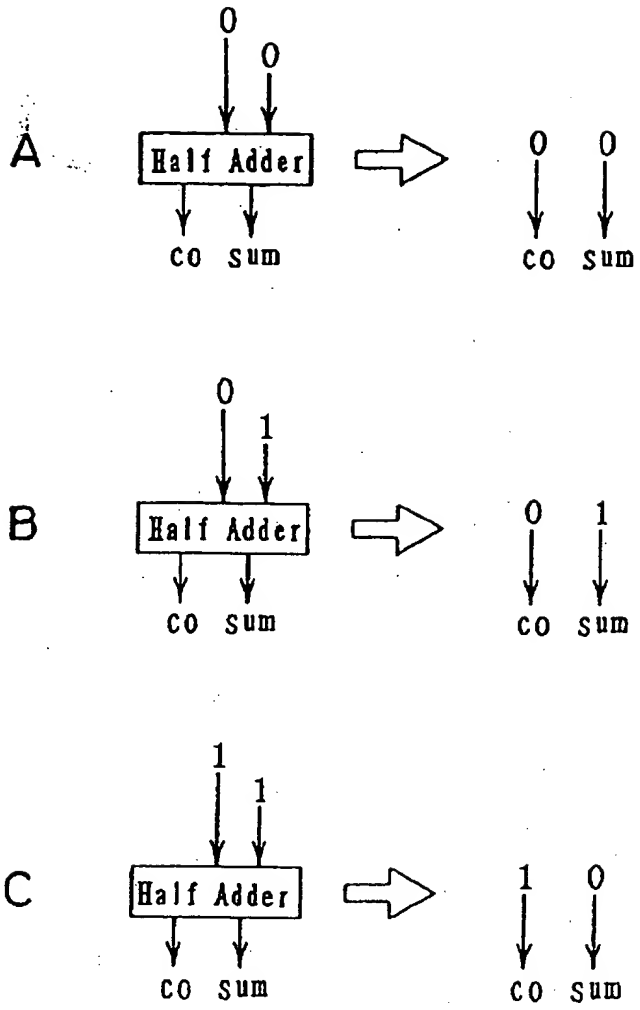
【図2】


A
0
0
0
Full Adder
CO
Sum
0
0
B
0
0
1
Full Adder
CO
Sum
0
1
C
0
1
1
Full Adder
CO
Sum
1
0
D
1
1
1
Full Adder
CO
Sum
1
1

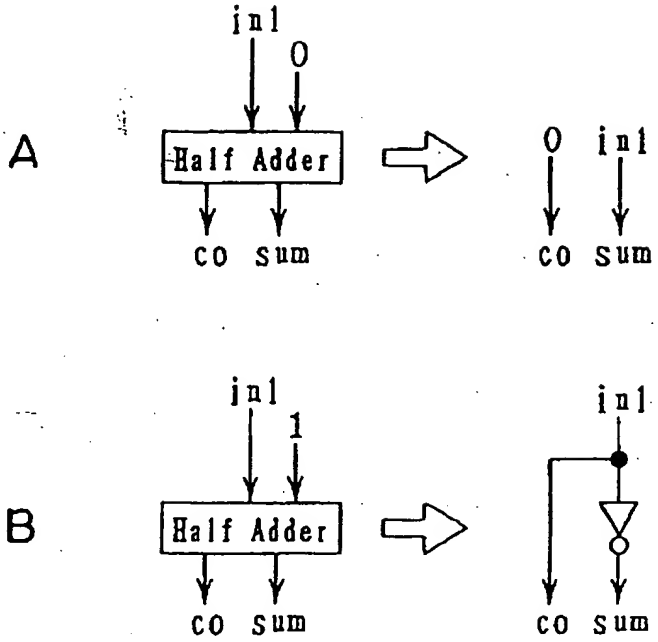
【図4】


A
in1
in2
0
Full Adder
CO
Sum
in1
in2
Half Adder
CO
Sum
B
in1
in2
1
Full Adder
CO
Sum
in1
in2
CO
Sum

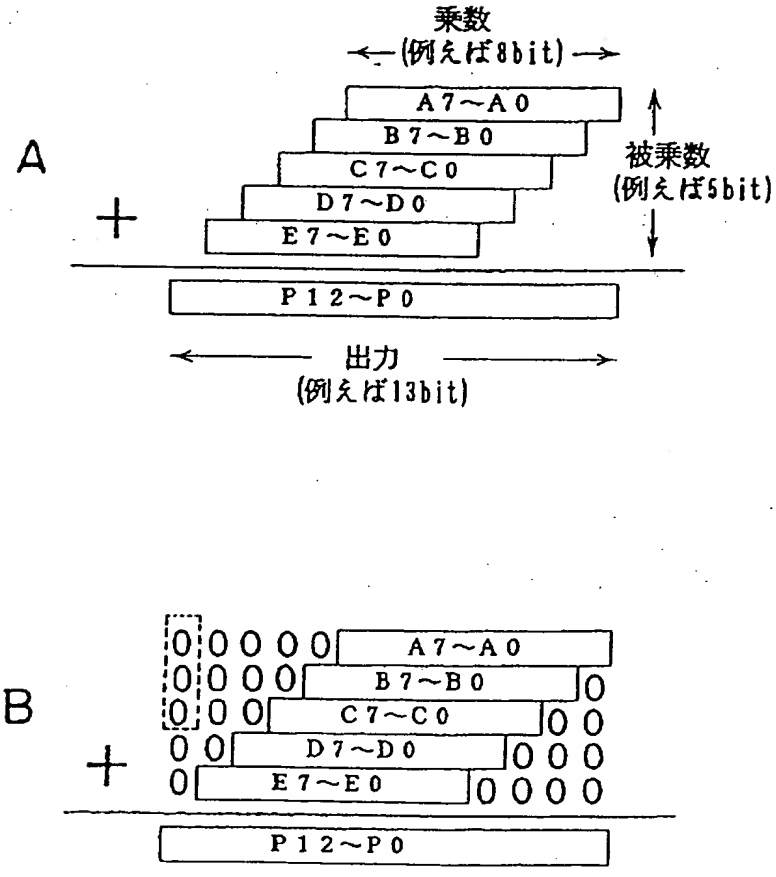
【図5】


A
0
0
Half Adder
CO
SUM
0
0
CO
SUM
B
0
1
Half Adder
CO
SUM
0
1
CO
SUM
C
1
1
Half Adder
CO
SUM
1
0
CO
SUM

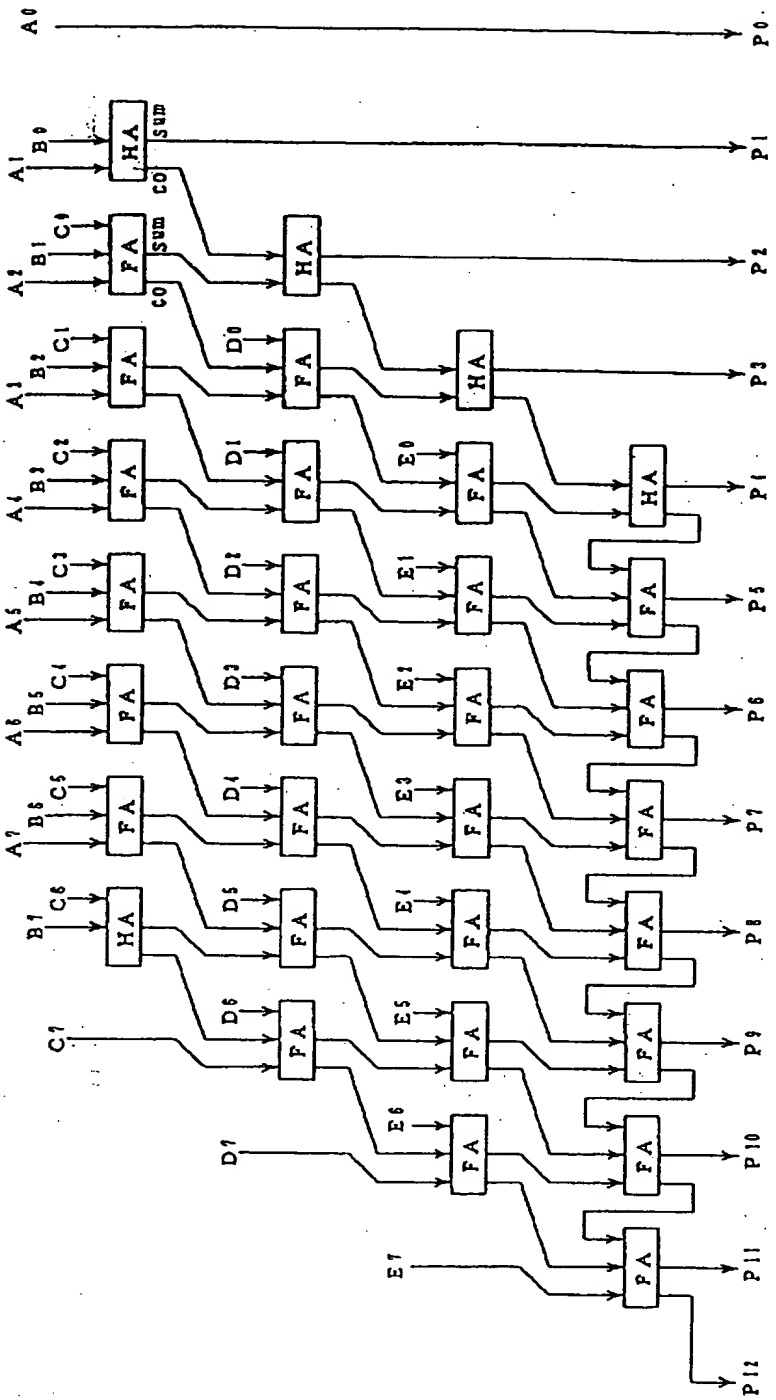
【図6】


A
in1
0
Half Adder
CO Sum
0 in1
CO Sum
B
in1
1
Half Adder
CO Sum
in1
CO Sum

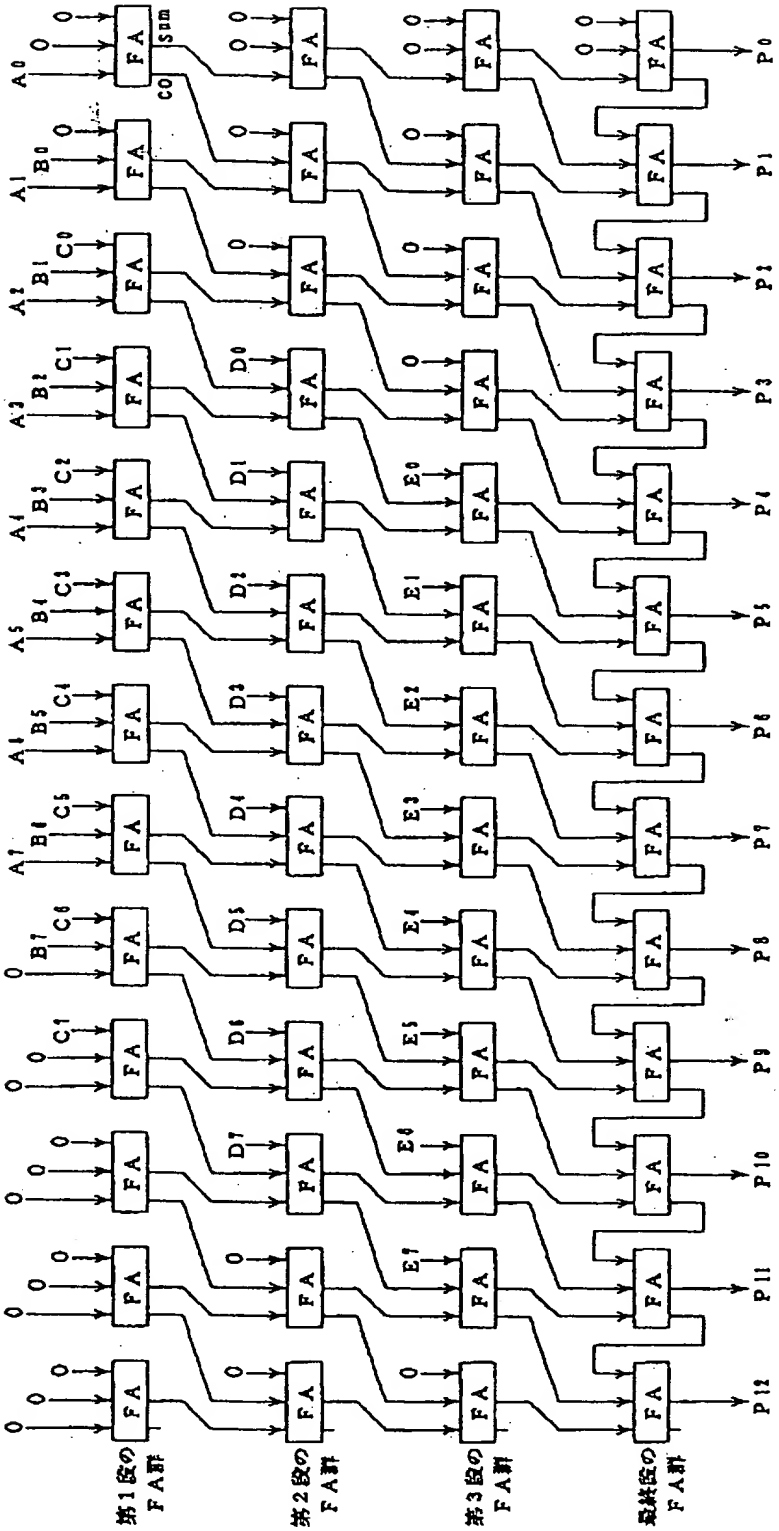
【図9】


乗数
(例えば8bit)
A
A7～A0
B7～B0
C7～C0
D7～D0
E7～E0
被乗数
(例えば5bit)
+
P12～P0
出力
(例えば13bit)
B
0 0 0 0 A7～A0
0 0 0 B7～B0 0
0 0 C7～C0 0 0
0 D7～D0 0 0 0
0 E7～E0 0 0 0 0
+
P12～P0

【図8】


A0
P0
A1
B0
HA
Sum
CO
P1
A2
B1
C0
FA
Sum
CO
HA
P2
A3
B2
C1
FA
D0
FA
HA
P3
A4
B3
C2
FA
D1
FA
E0
FA
HA
P4
A5
B4
C3
FA
D2
FA
E1
FA
FA
P5
A6
B5
C4
FA
D3
FA
E2
FA
FA
P6
A7
B6
C5
FA
D4
FA
E3
FA
FA
P7
B7
C6
HA
D5
FA
E4
FA
FA
P8
C7
D6
FA
E5
FA
FA
P9
D7
E6
FA
FA
P10
E7
FA
P11
P12

【図10】


FA
Sum
CO
第1段のFA群
第2段のFA群
第3段のFA群
最終段のFA群
P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12